

## LAB-8

### Creating a VPC in Terraform

**Step 1:** Create a main.tf file having all configuration

```
main.tf x
main.tf > resource "aws_subnet" "my_subnet"
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "5.35.0"
6     }
7   }
8 }
9 provider "aws" {
10   region = "ap-south-1"
11   access_key = " "
12   secret_key = " "
13 }
14 resource "aws_vpc" "my_vpc" {
15   cidr_block = "10.0.0.0/16"
16   enable_dns_support = true
17   enable_dns_hostnames = true
18   tags = {
19     Name = "MyVPC"
20   }
21 }
22 resource "aws_subnet" "my_subnet" {
23   count = 2
24   vpc_id = aws_vpc.my_vpc.id
25   cidr_block = "10.0.${count.index + 1}.0/24"
26   availability_zone = "ap-south-1a"
27   map_public_ip_on_launch = true
28   tags = {
29     Name = "MySubnet-${count.index + 1}"
30   }
31 }
```

## Step 2: Use terraform init to initialize terraform

```
> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.35.0"...
- Installing hashicorp/aws v5.35.0...
- Installed hashicorp/aws v5.35.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

🔍 ~/Desktop/spcmlab/lab8

## Step 3: Use terraform validate to check errors in HCL script

```
> terraform validate
Success! The configuration is valid.
```

🔍 ~/Desktop/spcmlab/lab8

## Step 4: Use terraform plan to verify given resources

```
> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_subnet.my_subnet[0] will be created
+ resource "aws_subnet" "my_subnet" {
  + arn                                = (known after apply)
  + assign_ipv6_address_on_creation    = false
  + availability_zone                  = "ap-south-1a"
  + availability_zone_id                = (known after apply)
  + cidr_block                          = "10.0.1.0/24"
  + enable_dns64                        = false
  + enable_resource_name_dns_a_record_on_launch = false
  + enable_resource_name_dns_aaaa_record_on_launch = false
  + id                                 = (known after apply)
  + ipv6_cidr_block_association_id      = (known after apply)
  + ipv6_native                         = false
  + map_public_ip_on_launch             = true
  + owner_id                           = (known after apply)
  + private_dns_hostname_type_on_launch = (known after apply)
  + tags                                = {
    + "Name" = "MySubnet-1"
  }
  + tags_all                            = {
    + "Name" = "MySubnet-1"
  }
  + vpc_id                              = (known after apply)
}
```

## Step 5: Use terraform apply to make changes in resources

```
> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_subnet.my_subnet[0] will be created
+ resource "aws_subnet" "my_subnet" {
  + arn                                = (known after apply)
  + assign_ipv6_address_on_creation    = false
  + availability_zone                  = "ap-south-1a"
  + availability_zone_id                = (known after apply)
  + cidr_block                          = "10.0.1.0/24"
  + enable_dns64                        = false
  + enable_resource_name_dns_a_record_on_launch = false
  + enable_resource_name_dns_aaaa_record_on_launch = false
  + id                                 = (known after apply)
  + ipv6_cidr_block_association_id      = (known after apply)
  + ipv6_native                         = false
  + map_public_ip_on_launch             = true
  + owner_id                           = (known after apply)
  + private_dns_hostname_type_on_launch = (known after apply)
  + tags                                = {
    + "Name" = "MySubnet-1"
  }
  + tags_all                            = {
    + "Name" = "MySubnet-1"
  }
  + vpc_id                              = (known after apply)
}
```

## Step 6: Verify it using AWS Console

The screenshot displays the AWS Management Console interface, specifically the VPC (Virtual Private Cloud) section. The top navigation bar includes the AWS logo, a search bar, and user information (Mumbai, ArnimTallyan). The left sidebar shows the 'Virtual private cloud' menu with options for 'Your VPCs', 'Subnets', 'Route tables', and 'Internet gateways'. The main content area is divided into two panels: 'Your VPCs (2)' and 'Subnets (5)'. The 'Your VPCs' panel shows a table with two VPCs, both in an 'Available' state. The 'Subnets' panel shows a table with five subnets, all in an 'Available' state. The subnets are associated with two VPCs: 'MyVPC' and 'vpc-0a6c28a2c5f810f0e'.

**Your VPCs (2)**

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
MyVPC	<a href="#">vpc-0899fa9083c9f9700</a>	Available	10.0.0.0/16	-
-	<a href="#">vpc-0a6c28a2c5f810f0e</a>	Available	172.31.0.0/16	-

**Subnets (5)**

Name	Subnet ID	State	VPC	IPv4 CIDR
MySubnet-2	<a href="#">subnet-05f0c085f9b7df2f4</a>	Available	<a href="#">vpc-0899fa9083c9f9700</a>   MyVPC	10.0.2.0/24
-	<a href="#">subnet-05726168eddb0c029</a>	Available	<a href="#">vpc-0a6c28a2c5f810f0e</a>	172.31.0.0/24
-	<a href="#">subnet-05d27f498c7f6a158</a>	Available	<a href="#">vpc-0a6c28a2c5f810f0e</a>	172.31.0.0/24
MySubnet-1	<a href="#">subnet-0900ae84902893c6e</a>	Available	<a href="#">vpc-0899fa9083c9f9700</a>   MyVPC	10.0.1.0/24
-	<a href="#">subnet-0055f9a760dbb70fc</a>	Available	<a href="#">vpc-0a6c28a2c5f810f0e</a>	172.31.0.0/24

## Step 7: Clear all resources by using terraform destroy

```
> terraform destroy
aws_vpc.my_vpc: Refreshing state... [id=vpc-0899fa9083c9f9700]
aws_subnet.my_subnet[0]: Refreshing state... [id=subnet-0900ae84902893c6e]
aws_subnet.my_subnet[1]: Refreshing state... [id=subnet-05f0c085f9b7df2f4]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_subnet.my_subnet[0] will be destroyed
- resource "aws_subnet" "my_subnet" {
  - arn                                = "arn:aws:ec2:ap-south-1:533266967718:subnet/subnet-0900ae84902893c6e" -> null
  - assign_ipv6_address_on_creation    = false -> null
  - availability_zone                  = "ap-south-1a" -> null
  - availability_zone_id                = "aps1-az1" -> null
  - cidr_block                         = "10.0.1.0/24" -> null
  - enable_dns64                       = false -> null
  - enable_lni_at_device_index         = 0 -> null
  - enable_resource_name_dns_a_record_on_launch = false -> null
  - enable_resource_name_dns_aaaa_record_on_launch = false -> null
  - id                                = "subnet-0900ae84902893c6e" -> null
  - ipv6_native                        = false -> null
  - map_customer_owned_ip_on_launch    = false -> null
  - map_public_ip_on_launch            = true -> null
  - owner_id                           = "533266967718" -> null
  - private_dns_hostname_type_on_launch = "ip-name" -> null
  - tags                               = {
    - "Name" = "MySubnet-1"
  }
```

## Step 8: Verify it by AWS Console

The screenshot displays the AWS Management Console interface. The top navigation bar includes the AWS logo, 'Services' menu, a search bar, and user information for 'Mumbai' and 'ArnimTaliyan'. The main content area is divided into two panels. The left panel shows the 'VPC dashboard' with a search bar and a 'Filter by VPC' dropdown. The right panel displays 'Your VPCs (1)' with a table containing one VPC: 'vpc-0a6c28a2c5f810f0e' in an 'Available' state with an IPv4 CIDR of '172.31.0.0/16'. Below this, the 'Subnets (3)' panel shows a table with three subnets, all in an 'Available' state, associated with the same VPC and having an IPv4 CIDR of '172.31.0.0/16'.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
-	<a href="#">vpc-0a6c28a2c5f810f0e</a>	Available	172.31.0.0/16	-

  

Name	Subnet ID	State	VPC	IPv4 CIDR
-	<a href="#">subnet-05726168eddb0c029</a>	Available	<a href="#">vpc-0a6c28a2c5f810f0e</a>	172.31.0.0/16
-	<a href="#">subnet-05d27f498c7f6a158</a>	Available	<a href="#">vpc-0a6c28a2c5f810f0e</a>	172.31.0.0/16
-	<a href="#">subnet-0055f9a760dbb70fc</a>	Available	<a href="#">vpc-0a6c28a2c5f810f0e</a>	172.31.0.0/16

**If you want make any changes in VPC modify main.tf and add desired resources and then use terraform apply to apply desired resources change.**